浙江大学

C++ 程 序 设 计

大 程 序 报 告

2018~2019 春夏学期　2019 年 6 月 12 日

报告撰写注意事项

1） 图文并茂。文字通顺，语言流畅，无错别字。

2） 书写格式规范，排版良好，内容完整。

3） 存在拼凑、剽窃等现象一律认定为抄袭；0 分

选题名称：飞机大战在 FLTK 下的实现

考评项		分数	说明
菜单系统		10	文件读取，图片读取，方向控制，游戏控制
图标工具栏		5	方向键 4 个，游戏控制键 2 个
快捷键		5	方向快捷键 4 个，文件读取快捷键 2 个，游戏控制快捷键 2 个
状态信息栏		5	右侧文本栏即时显示剩余子弹数与生命数
功能（35）	控制飞机	10	使用键盘快捷键或菜单栏或窗口方向键可以控制我方飞机前后左右运动，以及发射子弹
	随机发射	15	敌方三架飞机均随机运动，且随机发射子弹
	碰撞检测	10	我方子弹与敌方飞机碰撞，敌方飞机变红膨胀爆炸一秒。敌方子弹与我方飞机碰撞，我方飞机变红一秒且生命数减一。
链表		5	设计链表用于子弹序列的存储
文件	文件读取	10	保存剩余子弹数与剩余生命数到文本文件；从文件中读取图片作为玩家头像
	多文件	5	采用多文件组织的方式，使用了 13 个类
大程序报告	分析设计	5	软件简介，功能结构，全局、函数及重要算法说明，源程序中功能、函数、文件的组织关系
	部署运行	5	编译安装、运行测试、用户使用手册
	运行结果	5	效果展示
	分析	5	系统开发亮点和应用知识点总结

总分	100

1、题目描述和题目要求

题目描述：使用 FLTK 完成飞机大战的游戏实现。

题目要求：利用 FLTK 的 C++图形库，设计和实现一款带有图形界面的游戏。
要求基于 graphics_lib，设计和实现图形界面，至少运用以下技术和知识：
1）菜单、按钮、鼠标、键盘、文件；
2）体现 C++的内容；

2、需求分析

在作业中我们需要用到 FLTK 图形库，涉及到菜单栏、按键、鼠标、键盘等的回调函数的使用，文件、图片的读取、写入与显示。我们需要体现 C++d 内容，建立了许多功能各异的类，利用了 C++的继承性、多态性、封装性。

一个 FLTK 下的飞机大战小游戏需要实现什么功能呢？
就让我们从界面和操作两方面来分析。

在界面方面，我们需要一个黑色的地图背景，一架绿色的己方飞机，能控制发射绿色的己方子弹。多架（三架）黄色的敌方飞机，敌方飞机随机运动且随机发射黄色的敌方子弹。飞机被击中后，发生爆炸（变红膨胀）然后消失。我们可以使用 OpenGL 来绘制飞机与子弹，也可以使用 Fl_Box 来模拟，后者更利于随时改变位置与添加飞机图标。

操作方面，我们需要游戏控制键与菜单栏状态栏。游戏控制键包括 Start 键 Quit 键以开始和结束游戏，Stop 键暂停游戏，Font 键 Back 键 Left 键 Right 键控制己方飞机前进的方向，Start 键控制己方飞机发射子弹。菜单栏可以完成游戏控制键的所有功能，与从文件夹中读取图片显示在玩家头像上，以及将玩家得分情况保存为文本文件。文本状态栏显示当前剩余子弹数与当前得分情况，游戏开始时有 100 发子弹与 100 次生命，发射一颗子弹以及被敌机子弹击中一次会减少相应的数字。

3、总体设计

大致逻辑是，我在一个游戏窗口类(game_window)中绘制出游戏的主要构件如菜单栏状态栏控制按键等，使用游戏背景类(game_box)绘制游戏背景，有一个飞机绘制类(flyer_box)和一个子弹绘制类(bullet_draw_box)，编写己方飞机类(my_flyer_box)与敌方飞机类(enemy_flyer_box)继承飞机类，编写己方子弹类(my_bullet_draw)与敌方子弹类(enemy_bullet_draw)继承子弹类。有一个子弹位置

信息类(bullet_list_box)，编写己方子弹位置信息类(my_bullet_list)与敌方子弹位置信息类(enemy_bullet_list)继承位置信息类。此外，还有按键类(color_button)实现鼠标移动上去按键就变绿的功能，有图片导入类(image_box)来将选择图片导入为玩家头像。

3.1 功能模块设计

根据前文的分析，本游戏分为两大模块，游戏模块与应用模块。

游戏模块是为玩家提供游戏娱乐的模块。玩家通过按下界面右下方按键或使用键盘快捷键，控制屏幕上的己方飞机四处飞行与发射子弹，与随机飞行的敌方飞机交战。每发射一发子弹，文本状态栏里的 Bullet 减少 1。我方飞机被击中后会变红一秒，同时文本状态栏里的 Score 减少 1。敌方飞机被击中后变红同时膨胀，两秒后消失，然后再度刷新出来。子弹在飞跃边境或击中敌我飞机时会消失，位置信息也随之删除。敌方飞机在飞跃边境或者被我方飞机击中后，会随机从屏幕最上端刷新出来。

应用模块是实现文件与图片输入输出流的模块。包括文件模块与图片模块。图片模块让玩家按下菜单栏中的 Open，在弹出的选择窗口里选定自己喜欢的图片，就能将游戏界面右端的玩家头像修改为此图片。文件模块让玩家按下菜单栏中的 Save，就能将当下的游戏分数 Bullet 与 Score 以文本文档的格式保存在游戏主文件夹里。

此外还有一些小模块，如按下菜单栏中的 Help 会弹出游戏玩法介绍窗口的帮助模块等。

3.2 类结构设计

game_window类：游戏飞机大战的主窗口

```
class game_window : public Fl_Double_Window
{
public:
    static void btn_start_cb(Fl_Widget *w, void *data);
    static void btn_stop_cb(Fl_Widget *w, void *data);
    static void btn_font_cb(Fl_Widget *w, void *data);
    static void btn_back_cb(Fl_Widget *w, void *data);
    static void btn_left_cb(Fl_Widget *w, void *data);
    static void btn_right_cb(Fl_Widget *w, void *data);
    static void btn_close_cb(Fl_Widget *w, void *data);
    static void main_menu_cb(Fl_Widget *w, void *data);
    static void jpeg_show_cb(Fl_Widget *w, void *data);
    static void help_show_cb(Fl_Widget *w, void *data);
    void click_btn_show_jpeg(Fl_Widget *w, void *data);
    game_window(int w, int h, const char* title);
    ~game_window();
    //Fl_Double_Window *main_window;
    Fl_Menu_Bar *main_menu_bar;
```

```
    Fl_Text_Buffer *text_buff;
    Fl_Text_Display *text_dis;
    color_button *btn_start;
    color_button *btn_stop;
    color_button *btn_font;
    color_button *btn_back;
    color_button *btn_left;
    color_button *btn_right;
    color_button *btn_close;
    game_box *game_box1;
    //飞机
    my_flyer *my_flyer1;
    enemy_flyer *enemy_1;
    enemy_flyer *enemy_2;
    enemy_flyer *enemy_3;
    //子弹
    my_bullet_draw *my_bullet;
    enemy_bullet_draw *enemy_1_bullet_1;
    enemy_bullet_draw *enemy_1_bullet_2;
    enemy_bullet_draw *enemy_1_bullet_3;
    enemy_bullet_draw *enemy_2_bullet_1;
    enemy_bullet_draw *enemy_2_bullet_2;
    enemy_bullet_draw *enemy_2_bullet_3;
    enemy_bullet_draw *enemy_3_bullet_1;
    enemy_bullet_draw *enemy_3_bullet_2;
    enemy_bullet_draw *enemy_3_bullet_3;
    //读取文件夹中图片
    Fl_Image *jpeg_image;
    Fl_Box *jpeg_image_box;
    Fl_Shared_Image *jpeg_image_orig;
    //给敌机我机添加图标
    Fl_Image *jpeg_my_flyer;
    Fl_Image *jpeg_enemy1;
    Fl_Image *jpeg_enemy2;
    Fl_Image *jpeg_enemy3;
    Fl_Shared_Image *jpeg_flyer_box;
    //保存游戏信息到文本
    static std::ofstream outfile;
    //静态变量
    static int bullet_dre;//用来向main函数传递子弹的方向，等于此刻飞机的方向，存储在
my_bullet_pos中
    static my_bullet_list my_bullet_pos;//用来存储子弹们的位置与方向，每次刷新都改变数
组的数据，再交给draw绘制出来
    static enemy_bullet_list enemy_1_bullet_1_pos;
```

```
    static enemy_bullet_list enemy_1_bullet_2_pos;
    static enemy_bullet_list enemy_1_bullet_3_pos;
    static enemy_bullet_list enemy_2_bullet_1_pos;
    static enemy_bullet_list enemy_2_bullet_2_pos;
    static enemy_bullet_list enemy_2_bullet_3_pos;
    static enemy_bullet_list enemy_3_bullet_1_pos;
    static enemy_bullet_list enemy_3_bullet_2_pos;
    static enemy_bullet_list enemy_3_bullet_3_pos;
    static int flyer_dre;//用来向main函数传递方向键，按一次键就改变飞机的方向，然后清零
    static int use_time;//用来计数，每过0.05秒增加1
    static int btn_event;//用来向main函数传递方向键，目前除了start之外似乎没什么用，子弹方向可以由飞机方向得到
    static bool bullet_flag;//
    static int game_score;//用于游戏计分，被敌机击中减分
};
```

game_box 类：游戏飞机大战的黑色背景板

```
class game_box : public Fl_Box
{
public:
    game_box(int x, int y, int w, int h, const char *data);
    ~game_box();
};
```

color_button 类：实现鼠标悬停按键上方时改变颜色

```
class color_button : public Fl_Button
{
private:
    int handle(int e)
    {
        switch (e)
        {
        case FL_ENTER:
            color(FL_GREEN);
            labelsize(22);
            redraw();
            return 1;
        case FL_LEAVE:
            color(FL_GRAY);
            labelsize(20);
            redraw();
            return 1;
        default:
```

```
                return Fl_Button::handle(e);
            }
        }
public:
    color_button(int x, int y, int w, int h, const char*data);
    ~color_button();
};
```

flyer 类：敌我飞机总类

```
class flyer : public Fl_Box
{
public:
    flyer(int x, int y, int w, int h, const char *data);
    ~flyer();
    void move_font(int i);
    void move_back(int i);
    void move_left(int i);
    void move_right(int i);
};
```

my_flyer 类：我方飞机类

```
class my_flyer : public flyer
{
public:
    my_flyer(int x, int y, int w, int h, const char *data);
    ~my_flyer();
    int fight_flag = 0;
};
```

enemy_flyer 类：敌方飞机类

```
class enemy_flyer : public flyer
{
public:
    enemy_flyer(int x, int y, int w, int h, const char *data);
    ~enemy_flyer();
    //爆炸显示函数，完成变红与缩小功能
    void fighten(int counter);
    //fight_flag为0时初始化，为-1时正常随机运动，为1时爆炸效果
    int fight_flag = 0;
    //move_counter为计数器，为初始化与爆炸各自维持50次刷新的时间
    int move_counter = 0;
    //运动函数，通过fight_flag判断切换三种运动方式
    void enemy_flyer_move(int type);
};
```

bullet_draw 类：敌我子弹绘制总类

```
class bullet_draw
{
public:
    //初始化box数组
    bullet_draw();
    bullet_draw(int number);
    ~bullet_draw();
    Fl_Box **bullet;
};
```

my_bullet_draw 类：我方子弹绘制类

```
class my_bullet_draw : public bullet_draw
{
public:
    //初始化mine_bullet_box数组
    my_bullet_draw(int number);
    ~my_bullet_draw();
    //输入一个pos数组，根据数组中的x，y绘制相应的box数组
    void my_bullet_drawing(my_bullet_list my_bullet_pos);
    //敌机碰撞检测
    void my_bullet_collision(my_bullet_list my_bullet_pos, enemy_flyer *enemy_n);
};
```

enemy_bullet_draw 类：敌方子弹绘制类

```
class enemy_bullet_draw : public bullet_draw
{
public:
    //初始化enemy_bullet_box数组
    enemy_bullet_draw(int number);
    ~enemy_bullet_draw();
    //输入一个pos数组，根据数组中的x，y绘制相应的box数组(这个和my_bullet_draw是一样的，只是改了下颜色)
    void enemy_bullet_drawing(enemy_bullet_list enemy_bullet_pos);
    //我机碰撞检测
    void enemy_bullet_collision(enemy_bullet_list enemy_bullet_pos, my_flyer *my_plane);
};
```

bullet_list 类：敌我子弹位置信息总类（使用 deque 存储子弹位置信息）

```
class bullet_list
{
public:
    std::deque<int>listX;
```

```
    std::deque<int>listY;
    std::deque<int>listDre;

    bullet_list();
    ~bullet_list();
    //在数组最后插入数据
    void pushX(int x);
    void pushY(int y);
    void pushDre(int dre);
    //取出数组中的一个数据，序号从零开始
    int getX(int i);
    int getY(int i);
    int getDre(int i);
    //删除数组的一个元素
    void dele(int i);
    //返回数组长度与是否为空
    int length();
    bool isEmpty();
};
```

my_bullet_list 类：我方子弹位置信息类

```
class my_bullet_list : public bullet_list
{
public:
    my_bullet_list();
    ~my_bullet_list();
    //改变xy使得子弹向某个方向移动一格
    void change(int i);
};
```

enemy_bullet_list 敌方子弹位置信息类

```
class enemy_bullet_list : public bullet_list
{
public:
    enemy_bullet_list();
    ~enemy_bullet_list();
    //改变xy使得子弹纵向移动
    void enemy_bullet1(int i);
    //改变xy使得子弹斜向移动
    void enemy_bullet2(int i);
    void enemy_bullet3(int i);
};
```

image_box 类：图片导入类

```
class image_box : public Fl_Box
{
public:
    image_box(int x, int y, int w, int h, const char *data);
    ~image_box();
    void show_image(Fl_PNG_Image *pic);
    void show_image(Fl_JPEG_Image *pic);
};
```

3.3 函数功能描述

game_window类：游戏飞机大战的主窗口

```
class game_window : public Fl_Double_Window
{
game_window::game_window(int w, int h, const char *title) : Fl_Double_Window(w, h, title)
```

构造函数，构造一个 DoubleWindow 类对象，作为游戏主体窗口。
传入参数窗口的长、宽与标题。无返回值。
建立敌我飞机与子弹的 box，建立按键 button 与菜单栏 menu，建立文本状态栏与图片显示 box，分配调用函数与键盘快捷键。

```
game_window::~game_window()
```

析构函数，释放类指针。
无传入参数，无返回值。

```
void game_window::btn_start_cb(Fl_Widget *w, void *data)
```

开始按键回调函数，开始游戏以及发射子弹。
传入参数主窗口指针，无返回值。

```
void game_window::btn_stop_cb(Fl_Widget *w, void *data)
```

暂停按键回调函数，暂停游戏。
传入参数主窗口指针，无返回值。

```
void game_window::btn_font_cb(Fl_Widget *w, void *data)
void game_window::btn_back_cb(Fl_Widget *w, void *data)
void game_window::btn_left_cb(Fl_Widget *w, void *data)
void game_window::btn_right_cb(Fl_Widget *w, void *data)
```

移动按键回调函数，控制我机上下左右移动。
传入参数主窗口指针，无返回值。

```
void game_window::btn_close_cb(Fl_Widget *w, void *data)
```

退出按键回调函数，退出游戏。
传入参数主窗口指针，无返回值。

```
void game_window::main_menu_cb(Fl_Widget *w, void *data)
```

菜单栏回调函数，实现菜单栏的功能，包括控制我机移动与发射子弹，读取图片保存文本。
传入参数主窗口指针，无返回值。
对菜单栏操作进行判断，控制飞机的移动与子弹的发射，控制游戏的开始暂停与退出，调用帮助函数显示帮助窗口，调用图片读取函数与文件读写函数实现相应功能。

void game_window::help_show_cb(Fl_Widget *w, void *data)
菜单栏帮助界面回调函数，由菜单栏回调函数调用。
显示一个弹出窗口，一堆说明文字与两个选择键。
传入参数主窗口指针，无返回值。
使用 fl_ask 来建立有两个按键的帮助窗口。使用 fl_choice 来设置窗口显示帮助文字。

void game_window::jpeg_show_cb(Fl_Widget *w, void *data)
图片读取函数，由菜单栏回调函数调用。
选择图片，调用图片显示函数。
传入参数主窗口指针，无返回值。
使用 file_chooser 文件选择器来选择图片，建立 shared_image 对象读取图片。

void game_window::click_btn_show_jpeg(Fl_Widget *w, void *data)
图片显示函数，由图片读取函数调用。
将缓存中的图片显示在游戏主窗口右端玩家头像处。
传入参数主窗口指针，无返回值。
将缓存在 shared_image 中的图片复制到 image 类型的对象中，再添加入玩家头像 box 中。

game_box 类：游戏飞机大战的黑色背景板

class game_box : public Fl_Box

game_box::game_box(int x, int y, int w, int h, const char *data = 0) : Fl_Box(x, y, w, h, data)
构造函数，建立一个 Fl_Box 类对象，设置为窗口的黑色背景板。
无参数，无返回值。

game_box::~game_box()
析构函数，释放类指针。
无传入参数，无返回值。

color_button 类：实现鼠标悬停按键上方时改变颜色

class color_button : public Fl_Button

color_button::color_button(int x, int y, int w, int h, const char*data) : Fl_Button(x, y, w, h, data)
构造函数，建立一个 Fl_Button 类对象。
实现鼠标悬空选择按键。
传入参数按键的位置、长、宽与标题。无返回值。
使用鼠标 handle 事件，判断鼠标移动到按键上方时，将按键由灰色改为绿色。

color_button::~color_button()
析构函数，释放类指针。
无传入参数，无返回值。

flyer 类：敌我飞机总类

class flyer : public Fl_Box

flyer::flyer(int x, int y, int w, int h, const char *data = 0) : Fl_Box(x, y, w, h, data)

构造函数，建立一个 Fl_Box 类对象。

建立扁平 box 作为飞行物体的一般形态。

传入参数 box 的位置、长、宽与标题。无返回值。

flyer::~flyer()

析构函数，释放类指针。

无传入参数，无返回值。

void flyer::move_font(int i)

void flyer::move_back(int i)

void flyer::move_left(int i)

void flyer::move_right(int i)

飞行物体移动函数。

让飞行物体向四个方向移动。

传入参数移动的距离，无返回值。

调用 x、y 函数读取物体位置，hide 函数隐藏原来物体，resize 函数改变物体位置，redraw 函数重新绘制物体，show 函数让物体显示出来。实现让 box 对象向不同方向移动的功能。

my_flyer 类：我方飞机类

class my_flyer : public flyer

my_flyer::my_flyer(int x, int y, int w, int h, const char *data = 0) : flyer(x, y, w, h, data)

构造函数，建立一个 flyer 类对象。

传入参数为 flyer 的位置、长、宽与标题。无返回值。

设置我方飞机图标与颜色为绿色。

my_flyer::~my_flyer() {}

析构函数，释放类指针。

无传入参数，无返回值。

enemy_flyer 类：敌方飞机类

class enemy_flyer : public flyer

enemy_flyer::enemy_flyer(int x, int y, int w, int h, const char *data = 0) : flyer(x, y, w, h, data)

构造函数，建立一个 flyer 类对象。

传入参数为 flyer 的位置、长、宽与标题。无返回值。

设置敌方飞机图标与颜色为黄色。

enemy_flyer::~enemy_flyer()

析构函数，释放类指针。

无传入参数，无返回值。

void enemy_flyer::enemy_flyer_move(int type)
敌机移动函数。
让敌机随机向四个方向移动。
传入参数敌机的类型，无返回值。
判断敌机飞到边境后，返回随机初始位置。判断敌机被我机子弹击中后，调用 fighten 敌机击毁函数，然后返回随机初始位置。通过计数来让敌机循环执行前进一秒，随机运动两秒的动作。通过建立一个三位随机数，取个位十位百位为三种类型敌机的运动随机数，确保同一时间取到不同的三个随机数，敌机之间运动互相不影响。

void enemy_flyer::fighten(int counter)
敌机击毁函数。
让被击中的敌机变红膨胀一秒。
传入参数击毁时间，无返回值。
通过对时间进行计数，每隔 0.1 秒膨胀一点，直到时间截止则消失。

bullet_draw 类：敌我子弹绘制总类

class bullet_draw

bullet_draw::bullet_draw()
构造函数，建立一个 box 类对象。
无传入参数，无返回值。

bullet_draw::bullet_draw(int number)
重载构造函数，建立一个 box 类对象。
建立 box 指针数组来存储多个子弹。
传入参数子弹数目，无返回值。
设置子弹初始为白色扁平圆形，初始化位置为 0。

bullet_draw::~bullet_draw()
析构函数，释放类指针。
无传入参数，无返回值。

my_bullet_draw 类：我方子弹绘制类

class my_bullet_draw : public bullet_draw

my_bullet_draw::my_bullet_draw(int number)
重载构造函数，建立一个 bullet 对象。
建立 box 指针数组来存储多个我方子弹。
传入参数子弹数目，无返回值。
设置子弹初始为绿色扁平圆形，初始化位置为 0。

my_bullet_draw::~my_bullet_draw()
析构函数，释放类指针。
无传入参数，无返回值。

void my_bullet_draw::my_bullet_drawing(my_bullet_list my_bullet_pos)

我方子弹绘制函数，绘制我方子弹。
读取 list 中的子弹位置信息，绘制子弹。
传入参数我方子弹位置 list 类对象，无返回值。
使用 getX、getY 函数读取 list 中的我方子弹位置信息，使用 position 函数移动子弹，使用 hide、redraw、show 函数重新绘制子弹。

void my_bullet_draw::my_bullet_collision(my_bullet_list my_bullet_pos, enemy_flyer *enemy_n)
敌机碰撞检测函数。
检测我方飞机发射的子弹是否击中敌机。
传入参数我方子弹位置list类对象，敌机类对象，无返回值。
使用getX、getY函数读取我方子弹位置信息，使用x、y函数读取敌机位置信息，判断是否碰撞，若碰撞，将敌机类对象中的flag置1。

enemy_bullet_draw 类：敌方子弹绘制类

class enemy_bullet_draw : public bullet_draw

enemy_bullet_draw::enemy_bullet_draw(int number)
重载构造函数，建立一个 bullet 对象。
建立 box 指针数组来存储多个敌方子弹。
传入参数子弹数目，无返回值。
设置子弹初始为黄色扁平圆形，初始化位置为 0。

enemy_bullet_draw::~enemy_bullet_draw()
析构函数，释放类指针。
无传入参数，无返回值。

void enemy_bullet_draw::enemy_bullet_drawing(enemy_bullet_list enemy_bullet_pos)
敌方子弹绘制函数，绘制敌方子弹。
读取 list 中的子弹位置信息，绘制子弹。
传入参数敌方子弹位置 list 类对象，无返回值。
使用 getX、getY 函数读取 list 中的敌方子弹位置信息，使用 position 函数移动子弹，使用 hide、redraw、show 函数重新绘制子弹。

void enemy_bullet_draw::enemy_bullet_collision(enemy_bullet_list enemy_bullet_pos, my_flyer *my_plane)
我机碰撞检测函数。
检测敌方飞机发射的子弹是否击中我机。
传入参数敌方子弹位置list类对象，我机类对象，无返回值。
使用getX、getY函数读取敌方子弹位置信息，使用x、y函数读取我机位置信息，判断是否碰撞，若碰撞，将我机类对象中的flag置1。

bullet_list 类：敌我子弹位置信息总类（使用 deque 存储子弹位置信息）

class bullet_list

bullet_list::bullet_list()

构造函数，建立一个list类对象。
无传入参数，无返回值。

bullet_list::~bullet_list()
析构函数，释放类指针。
无传入参数，无返回值。

void bullet_list::pushX(int x)
void bullet_list::pushY(int y)
void bullet_list::pushDre(int dre)
赋值函数。
向list类对象中的x，y，dre三个deque赋值。
传入参数待赋的值，无返回值。
调用了push_back函数，在deque的末尾添加元素。

int bullet_list::getX(int i)
int bullet_list::getY(int i)
int bullet_list::getDre(int i)
取值函数。
从list类对象中的x，y，dre三个deque中取指定位置的值。
传入待取值的序号，无返回值。
直接利用方括号引用deque中的元素。

void bullet_list::dele(int i)
删除函数。
将list类对象中的x，y，dre三个deque中删除首位元素。
传入删除元素数目，无返回值。
调用了pop_front函数，将deque的首位元素删除。

int bullet_list::length()
长度函数，计算list类对象的长度。
无传入参数，无返回值。
调用了size函数，取deque的长度值。

bool bullet_list::isEmpty()
判空函数，判断list类对象是否为空。
无传入参数，无返回值。
调用了empty函数，判断deque是否为空。

my_bullet_list 类：我方子弹位置信息类
class my_bullet_list : public bullet_list
my_bullet_list::my_bullet_list()
构造函数，建立一个list类对象。
无传入参数，无返回值。

my_bullet_list::~my_bullet_list()
析构函数，释放类指针。
无传入参数，无返回值。

void my_bullet_list::change(int i)
我方子弹移动函数。
修改我方子弹位置信息。
传入参数移动的方向，无返回值。
将我方子弹的位置信息修改，达到移动的效果。

enemy_bullet_list 敌方子弹位置信息类
class enemy_bullet_list : public bullet_list
enemy_bullet_list::enemy_bullet_list()
构造函数，建立一个list类对象。
无传入参数，无返回值。

enemy_bullet_list::~enemy_bullet_list()
析构函数，释放类指针。
无传入参数，无返回值。

void enemy_bullet_list::enemy_bullet1(int i)
void enemy_bullet_list::enemy_bullet1(int i)
void enemy_bullet_list::enemy_bullet1(int i)
敌方子弹移动函数。
修改敌方子弹位置信息。
传入参数移动的方向，无返回值。
将敌方子弹的位置信息修改，达到移动的效果。分为三个函数，使得三种类型敌机的子弹的飞行轨迹各不相同。

image_box 类：图片导入类
class image_box : public Fl_Box
image_box::image_box(int x, int y, int w, int h, const char *data = 0) : Fl_Box(x, y, w, h, data)
构造函数，建立一个box类对象。
传入参数box的位置、长、宽与标题，无返回值。
建立一个用来显示玩家头像的box类对象。

void image_box::show_image(Fl_PNG_Image *pic)
图片读取函数，用以读取PNG类型的图片。
传入参数图片类指针，无返回值。

void image_box::show_image(Fl_JPEG_Image *pic)
图片读取函数，用以读取JPEG类型的图片。
传入参数图片类指针，无返回值。

main：主体程序

void time_callback(void* game)

时钟回调函数。

每隔0.05秒执行一次，实现屏幕的刷新。

传入window类指针，无返回值。

实现start按键控制我机子弹发射，实现敌机子弹随机发射，实现方向键控制我机前后左右移动，实现敌机随机移动，实现暂停游戏与开始游戏，实现我机与敌机被击中的效果，实现碰撞检测函数随时调用，实现文本状态栏显示子弹剩余数目与生命剩余数量。

int main()

主程序，建立 window 类对象。

无传入参数，无返回值。

反复执行 time_callback 函数。

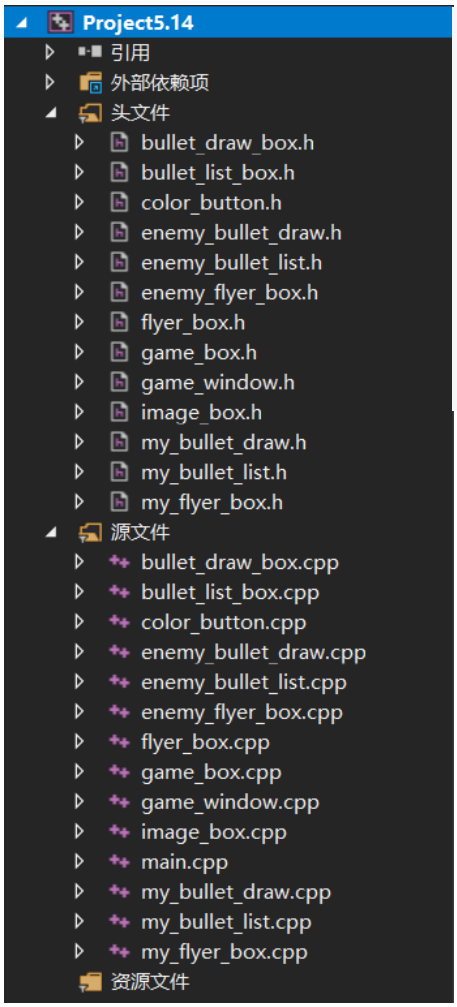
（每个函数按如下形式描述：

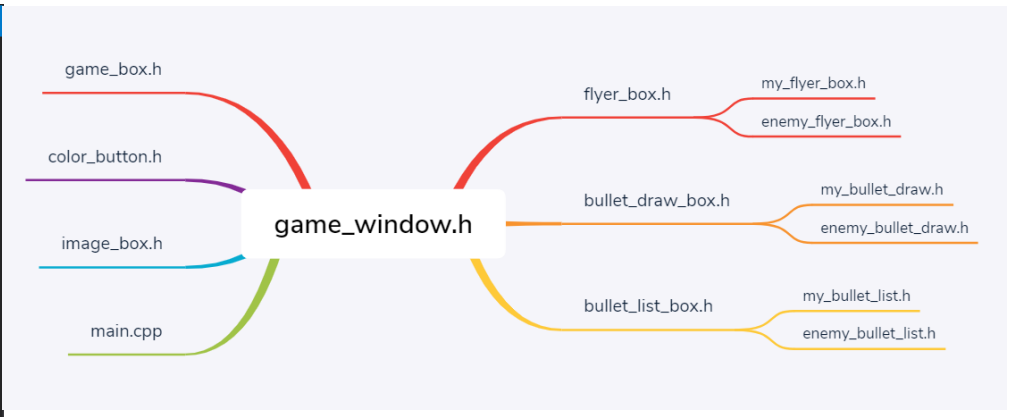
函数原型：　　功能描述；　　参数描述：　　返回值描述：　　函数算法描述：）

3.4 程序文件结构

1）文件函数结构

（可通过图或文字描述程序分为几个文件（需写出文件全名，即*.h，*.cpp）,每个.cpp 文件包含哪些函数定义，每个.h 包含哪些内容。）





◆ game_window.h 包含类 game_window 的声明，主界面上控制按键、菜单栏、文本状态栏、敌我飞机与子弹、各种回调函数的声明，构造函数与析构函数的声明。

◆ game_window.cpp 包含构造函数与析构函数的定义，与各种回调函数的定义，例如按键回调函数，菜单栏回调函数，图片读取回调函数，文本保存回调函数。

◆ game_box.h 包含类 game_box 的声明，构造函数与析构函数的声明。

◆ game_box.cpp 包含构造函数与析构函数的定义。

◆ color_button.h 包含类 color_button 的声明，构造函数与析构函数的声明。

◆ color_button.cpp 包含构造函数与析构函数的定义，鼠标 handle 事件的判断。

◆ image_box.h 包含类 image_box 的声明。构造函数与析构函数的声明。

◆ image_box.cpp 包含构造函数与析构函数的定义。

◆ main.cpp 包含

◆ flyer_box.h 包含 flyer 类的声明，包含 move 移动函数的声明，包含构造函数与析构函数的声明。

◆ flyer_box.cpp 包含包含移动函数的定义，构造函数与析构函数的定义。

◆ my_ flyer_box.h 包含类 my_flyer 的声明，flag 标志的声明，构造函数与析构函数的声明。

◆ my_ flyer_box.cpp 包含构造函数与析构函数的定义。

◆ enemy_ flyer_box.h 包含类 enemy_flyer 的声明，flag 标志的声明，fighten 被击中函数的声明，enemy_flyer_move 敌机移动函数的声明，构造函数与析构函数的声明。

◆ enemy_ flyer_box.cpp 包含 fighten 被击中函数的定义，enemy_flyer_move 敌机移动函数的定义，构造函数与析构函数的定义。

◆ bullet_draw_box.h 包含类 bullet_draw 的声明，指针数组 bullet 的声明，构造函数与析构函数的声明。

◆ bullet_draw_box.cpp 包含构造函数与析构函数的定义。

◆ my_ bullet_draw.h 包含类 my_bullet_draw 的声明，my_bullet_drawing 我方子弹绘

制函数的声明，my_bullet_collision 敌机碰撞检测函数的声明，构造函数与析构函数的声明。

- my_ bullet_draw.cpp 包含 my_bullet_drawing 我方子弹绘制函数的定义，my_bullet_collision 敌机碰撞检测函数的定义，构造函数与析构函数的定义。
- enemy_ bullet_draw.h 包含类 enemy_bullet_draw 的声明，enemy_bullet_drawing 敌方子弹绘制函数的声明，enemy_bullet_collision 我机碰撞检测函数的声明，构造函数与析构函数的声明。
- enemy_ bullet_draw.cpp 包含 enemy_bullet_drawing 敌方子弹绘制函数的定义，enemy_bullet_collision 我机碰撞检测函数的定义，构造函数与析构函数的定义。
- bullet_list_box.h 包含类 bullet_list 的声明，push 插入函数与 get 取出函数的声明，dele 删除函数与 length 取长度函数与 isEmpty 判空函数的声明，构造函数与析构函数的声明。
- bullet_list_box.cpp 包含 push 插入函数与 get 取出函数的定义，dele 删除函数与 length 取长度函数与 isEmpty 判空函数的定义，构造函数与析构函数的定义。
- my_bullet_list.h 包含类 my_bullet_list 的声明，change 子弹移动函数的声明，构造函数与析构函数的声明。
- my_ bullet_list.cpp 包含 change 子弹移动函数的定义，构造函数与析构函数的定义。
- enemy_ bullet_list.h 包含类 enemy_bullet_list 的声明，bullet1 子弹移动函数与 bullet2 子弹移动函数与 bullet3 子弹移动函数的声明，构造函数与析构函数的声明。
- enemy_ bullet_list.cpp 包含 bullet1 子弹移动函数与 bullet2 子弹移动函数与 bullet3 子弹移动函数的定义，构造函数与析构函数的定义。

2）多文件构成机制

（说明分文件构成程序的实现机制）

main.cpp中使用Fl::add_timeout(1.0, time_callback, window);　return Fl::run();语句循环调用time_callback函数，因而只需包含game_window.h。而game_window.h将game_box.h、color_button.h、my_flyer_box.h、enemy_flyer_box.h、my_bullet_draw.h、my_bullet_list.h、enemy_bullet_draw.h、enemy_bullet_list.h全都包含进来，实现主要的游戏功能。其中my_bullet_draw.h、enemy_bullet_draw.h又包含了bullet_draw_box.h，my_bullet_list.h、enemy_bullet_list.h又包含了bullet_list_box.h。

4.运行结果

4.1　编译安装运行说明

（此部分介绍如何由提交的源代码包，进行存放、编译生成.exe 文件的过程说明，以及运行.exe 后的用户使用手册。）

解压源代码压缩包，得到文件夹 game，使用 visual studio 打开其目录下的

Project5.14.sln 文件，将编译模式从 x64 修改为 x86，编译运行。如此能在 game 文件夹内的 Debug 目录下得到 Project5.14.exe 文件，点击即可开始游戏。

屏幕左半为游戏界面，你可以在这个界面里进攻敌方飞机。屏幕右上为游戏状态，显示您剩余的子弹数与剩余生命数。游戏开始阶段您一共有100发子弹与100条生命。屏幕右中为玩家头像，您可以在菜单栏中的Open打开图片修改头像。屏幕右下为控制按键，与菜单栏按键一样，Save存储游戏得分，Quit退出游戏，Open修改玩家头像。Start开始游戏并控制玩家飞机发射子弹，Stop暂停游戏，Font控制玩家飞机前进，Back控制玩家飞机后退，Left控制玩家飞机左移，Right控制玩家飞机右移。您可以在游戏界面的菜单栏里的Help打开帮助界面。

4.2 典型测试情况

（选取测试阶段典型的案例，说明如何设计测试数据，发现和定位错误的，测试结果可以含有屏幕截图。）

- 类的交叉调用，A 类调用 B 类，B 类也要调用 A 类。这样会出错。如何解决呢？方法一：不使用 include 语句，而是在 class A{}前面添加 class B；在 class B{}前面添加 class A；分别声明一下即可。方法二：在.h 文件里尽量不互相调用，要调用也只调用一方的。然后在.cpp 文件里就可以随便调用啦。
- 子弹显示问题，一开始把子弹写在了 game_window 里，我方飞机可以随按键控制而移动，子弹若是同样写法，就会计算完位置再 redraw 出来，也就是瞬移了。这样不行，得在 main 函数里调用 time_callback 函数，每隔 0.05 秒调用一次，再把子弹的 redraw 写在 time_callback 函数里，这样就能逐步移动子弹了。
- 子弹序列的使用，一开始我建立了一个模板类，然后自己写了一个链表。后来发现过于麻烦，我并不需要链表的从中间插入元素的功能。于是我改用 vector，但是 vector 是个单向的动态数组，无法删除首位元素，于是我使用 swap 将首尾交换了再删除，这样还是很麻烦。于是我改用 list，但 list 是双向链表，占用空间大。于是我再度改为 deque，deque 是双向动态数组，能够从尾部添加元素也能从首位删除元素，满足我的需求。
- 使用随机数来表示三个敌机的飞行状态。但若是同时生成三个随机数，由于随机数函数是根据系统时间生成的，会生成三个一样的随机数。解决方法可以是采用第一个随机数作为种子生成第二个随机数，再作种子生成第三个随机数。但这样很麻烦。而我生成一个三位的随机数，分别取个位、十位、百位，经过运算限制在一定范围内，就能得到三个不同的随机数了。
- 我方飞机被敌方子弹击中时，应该变红一秒然后再恢复黄色。这个如何完成?整个 time_callback 函数都在不断刷新，若是使用 int 变量计数，会导致变量初始化反复执行。要么在函数外定义全局变量，要么定义静态变量，这样都会占用程序的内存空间。再或者，定义一个新的变量存储前次的 flag，使得函数刷新时变量改变只执行一次。但是这样也很麻烦，我得引入两个变量，两个循环。一个循环判断是否被击中且只响应一次，一个循环

按照时间执行五十次退出。最后我把变量初始化放在 flag=0 时，在 flag=1 时让变量++，当变量为 50 时跳出，再初始化。如此就能满足在函数不断刷新的时候计时一秒了。

- 在存储文本数据的时候发生闪退，一行一行注释，每行都输出打印错误信息，上网必应搜索才发现，若是把非静态变量赋值给 file，有时候会发生错误。只能是传递静态变量了。
- 在编写图片读取与显示在屏幕右端 box 里面的时候，又遇到莫名其妙的闪退。反反复复看了许多次代码，都没有问题。调试发现一个指针变量为空指针。最后才发现，不是读取图片的语句出了问题，而是调用“调用这段语句的函数”的函数的一个指针传递错了，进而导致指针信息没有传递到位，我使用空指针去绘制 box 自然会闪退了。教训是，指针不能乱指，一定要思考清楚指针所指向的位置。

5.总结

7.1 程序亮点或创新之处

本程序在屏幕上绘制飞机与子弹时无闪烁。Open 可以便捷修改玩家头像。Save 可以存储玩家游戏得分。子弹信息类为自己写的类数组，其中的每个元素都含有三个 int 型变量，分别表示位置与方向。灵活利用了 C++的继承性，敌方子弹类我方子弹类都继承了子弹总类，敌方飞机类我方飞机类都继承了飞机总类，二者又都继承了飞行物体类。灵活利用了 C++的多态性，多次重载了移动函数。

7.2 应用知识点

C++的继承性、多态性的使用，我写的各个类大多继承于 Fl_Window 类与 Fl_Box 类，重载了许多函数。

FLTK 库中的 Fl_Window、Fl_Box、Fl_Button、Fl_Menu_Bar、Fl_Image、Fl_Text 类及其成员函数的使用。以及使用了一些特殊的函数，如计时器函数，新建窗口函数，窗口选择器函数等等。

7.3 心得体会

通过这次大作业，我初步了解了 C++图形库的基本知识，学习了 STL 库中容器的使用，学习了 FLTK 库中 UI 界面的设计与图形界面的实现，为今后学习使用 QT 等图形库打下一定基础。

此外，对 C++的理解也得到了加深。能够运用封装性、多态性、继承性以及类模板、函数重载来实现自己想要的功能，提升了代码编写能力、多文件协调能力与调试 debug 能力。

6、参考文献和资料

（列出参考的书籍、论文、网站的信息和地址等。）

【1】FLTK 官方教程：FLTK 中的小部件

https://www.fltk.org/doc-1.3/common.html

【2】FLTK 官方教程：在 FLTK 中绘制内容

https://www.fltk.org/doc-1.3/drawing.html#sect_WhenCanYouDraw

【3】C++中 STL 用法超详细总结

https://blog.csdn.net/u010183728/article/details/81913729

【4】什么是虚函数？虚函数的作用和使用方法

https://www.cnblogs.com/jianyungsun/p/6361670.html

【5】Erco's FLTK Cheat Page

http://seriss.com/people/erco/fltk/#Fl_Text_Display

【6】C++ 实现链表类（链表的各种操作）

https://blog.csdn.net/zhouzzz000/article/details/80627428